🚀 Engunity AI

Complete Project Architecture & Structure

AI-Powered SaaS Platform for Research, Code Generation & Data Analysis

📄 Download as PDF

📋 Project Overview

Engunity AI is a comprehensive AI-powered SaaS platform designed for research assistance, code generation, and data analysis. Built with scalability, modularity, and cost-efficiency in mind, it combines cutting-edge AI capabilities with blockchain security extensions.

Key Objectives

- **Scalable Architecture:** Modular design supporting horizontal scaling
- **AI-First Approach:** Integration of both cloud (Groq) and local (Phi-2) LLMs
- **Cost Efficiency:** Leveraging free-tier services for initial deployment
- **Security Focus:** Blockchain integration for content provenance and identity
- **Developer Friendly:** Clear structure for team collaboration

🧠 Core Features

Primary Features

1. **AI Chatbot (Groq + Phi-2 Fallback)** - Responds to English queries using Groq API, falls back to local Phi-2
2. **Code Assistant** - Generates, debugs, and explains Python code with best-practice suggestions
3. **Document Q&A** - Upload PDFs/DOCs and ask questions; uses RAG with FAISS for context-aware answers
4. **Research Writing Support** - Summarization, paraphrasing, section writing, citation formatting (APA/IEEE)
5. **Notebook Interface** - Monaco editor with AI help for code editing, execution, and saving
6. **Data Analysis Module** - Upload CSV/Excel, perform EDA, visualize trends with AI help
7. **Secure Code Execution** - Run user-submitted code safely in Docker sandbox with resource limits
8. **User Authentication** - Firebase Auth integration
9. **File Storage** - Secure upload/download via AWS S3 or Supabase Storage
10. **Session History** - Store/retrieve chat, notebook, and file history (PostgreSQL/DynamoDB)

Extended Features

1. **Project Planner** - AI-based milestone planner + Kanban task board
2. **Citation Manager** - BibTeX, DOI support, auto-generated references
3. **Version Control** - Save/revert notebooks, research drafts, and chat logs
4. **Literature Review Assistant** - Summarize multiple papers, auto-highlight gaps
5. **Table & Chart Interpreter** - Explain uploaded tables/graphs using natural language
6. **AI Agents (Local)** - LangChain/MiniChain tools for code review, PDF summaries, task decomposition
7. **Prompt Templates Gallery** - Templates for FastAPI, datasets, paper summaries, etc.
8. **GitHub Integration Tools** - Analyze issues, suggest fixes, generate docs
9. **Job Prep Toolkit** - Resume review, mock interviews, skill gap insights

10. **Freemium Tier Support** - Role-based access (Free/Pro/Edu)

Blockchain + AI Security Extensions

1. **Decentralized AI Marketplace Integration** - Browse, query, and pay for external AI models via smart contracts
2. **Smart Contract Auditor AI Agent** - Upload and analyze Solidity contracts for vulnerabilities
3. **Blockchain-Provenance Content Generator** - Register AI-generated output on blockchain
4. **Secure Identity via DID + AI** - Blockchain-based login using Decentralized Identity
5. **Secure Blockchain Logging + Anomaly Detector** - Log sensitive activity to private blockchain
6. **Credential Verification + Adaptive AI Learning** - Issue on-chain certificates for completions

🏗️ Complete Directory Structure

```
engunity-ai/
├── .github/                          # GitHub configurations
│   ├── workflows/                    # CI/CD pipelines
│   │   ├── frontend-deploy.yml       # Frontend deployment to Vercel
│   │   ├── backend-deploy.yml        # Backend deployment to Railway
│   │   ├── test-suite.yml           # Automated testing pipeline
│   │   └── security-scan.yml        # Security vulnerability scanning
│   ├── ISSUE_TEMPLATE/              # Issue templates for bug/feature re
│   ├── PULL_REQUEST_TEMPLATE.md     # PR template with checklist
│   └── dependabot.yml               # Automated dependency updates
│
├── frontend/                         # Next.js frontend application
│   ├── src/
│   │   ├── app/                     # Next.js 14 app directory
│   │   │   ├── (auth)/              # Authentication routes group
│   │   │   │   ├── login/           # Login page
│   │   │   │   ├── register/       # Registration page
│   │   │   │   └── forgot-password/ # Password recovery
│   │   │   ├── (dashboard)/         # Protected dashboard routes
│   │   │   │   ├── layout.tsx       # Dashboard layout wrapper
│   │   │   │   ├── page.tsx         # Dashboard home
│   │   │   │   ├── chat/           # AI chatbot interface
│   │   │   │   ├── code/           # Code assistant
│   │   │   │   ├── documents/      # Document Q&A
│   │   │   │   ├── research/       # Research writing tools
│   │   │   │   ├── notebook/       # Code notebook interface
│   │   │   │   ├── analysis/       # Data analysis module
│   │   │   │   ├── projects/       # Project planner
│   │   │   │   ├── citations/      # Citation manager
│   │   │   │   ├── marketplace/    # AI marketplace (Web3)
│   │   │   │   ├── audit/          # Smart contract auditor
│   │   │   │   └── settings/       # User settings
│   │   │   ├── api/                # API routes for server actions
│   │   │   └── layout.tsx           # Root layout
│   │   ├── components/              # Reusable React components
│   │   │   ├── ui/                 # ShadCN UI components
│   │   │   ├── chat/               # Chat-specific components
│   │   │   ├── editor/             # Monaco editor components
│   │   │   ├── analysis/           # Data viz components
│   │   │   ├── auth/               # Auth components
│   │   │   ├── blockchain/         # Web3 components
│   │   │   └── shared/             # Shared components
│   │   ├── hooks/                  # Custom React hooks
│   │   ├── lib/                    # Utility functions
│   │   ├── store/                  # Zustand state management
│   │   ├── types/                  # TypeScript type definitions
│   │   └── styles/                 # Global styles and Tailwind
│   ├── public/                     # Static assets
│   ├── tests/                      # Frontend tests
│   └── [config files]              # Next.js, Tailwind, TS configs
```

```
│
├── backend/                          # FastAPI backend application
│   ├── app/
│   │   ├── api/                     # API endpoints
│   │   │   └── v1/                  # API version 1
│   │   ├── core/                    # Core configurations
│   │   ├── models/                  # Data models
│   │   ├── services/                # Business logic services
│   │   ├── agents/                  # LangChain agents
│   │   ├── middleware/              # FastAPI middleware
│   │   ├── utils/                   # Utility functions
│   │   └── main.py                  # FastAPI app entry
│   ├── vector_store/                # FAISS vector database
│   ├── sandbox/                     # Docker sandbox configs
│   ├── migrations/                  # Database migrations
│   ├── tests/                       # Backend tests
│   └── [config files]               # Requirements, Dockerfile
│
├── blockchain/                       # Blockchain components
│   ├── contracts/                   # Smart contracts
│   ├── scripts/                     # Deployment scripts
│   ├── test/                        # Contract tests
│   └── hardhat.config.js            # Hardhat configuration
│
├── infrastructure/                   # Infrastructure as Code
│   ├── docker/                      # Docker configurations
│   ├── kubernetes/                  # K8s manifests (future)
│   └── terraform/                   # Cloud infrastructure
│
├── scripts/                         # Development scripts
│   ├── setup/                       # Setup scripts
│   ├── dev/                         # Development scripts
│   └── deploy/                      # Deployment scripts
│
├── docs/                            # Documentation
├── data/                            # Data directory
├── config/                          # Configuration files
└── [root config files]              # Docker, Git, Lint configs
```

📦 Module Breakdown

Frontend Modules

1. Authentication Module

- Location: `frontend/src/app/(auth)/`
- User registration with email verification
- Social login integration (Google, GitHub)
- Password reset functionality
- Multi-factor authentication support

2. Chat Module

- Location: `frontend/src/app/(dashboard)/chat/`
- Real-time chat interface with streaming responses
- Conversation history and search
- Code syntax highlighting
- File attachment support

3. Code Assistant Module

- Location: `frontend/src/app/(dashboard)/code/`
- Code generation interface
- Debugging assistant
- Best practices suggestions
- Language-specific templates

4. Document Q&A Module

- Location: `frontend/src/app/(dashboard)/documents/`
- File upload interface (PDF, DOCX, TXT)
- Document viewer with annotations
- Question interface with context highlighting
- Document management dashboard

5. Research Module

- Location: `frontend/src/app/(dashboard)/research/`
- Paper summarization interface
- Citation formatter
- Literature review assistant
- Research template gallery

6. Notebook Module

- Location: `frontend/src/app/(dashboard)/notebook/`
- Monaco editor integration
- Multi-language support
- Real-time collaboration features
- Version control interface

Backend Services

1. AI Service

- Location: `backend/app/services/ai/`
- Groq API integration with retry logic
- Local Phi-2 model management
- Request routing and fallback logic
- Response caching and optimization

2. Document Service

- Location: `backend/app/services/document/`
- PDF parsing and extraction
- Text chunking and preprocessing
- Vector embedding generation
- RAG implementation with FAISS

3. Code Execution Service

- Location: `backend/app/services/code/`
- Docker sandbox management
- Resource limitation enforcement
- Output streaming

- Security scanning

4. Blockchain Service

- Location: `backend/app/services/blockchain/`
- Web3 provider management
- Smart contract interactions
- Transaction monitoring
- Gas optimization

Agent System

1. Research Agent

- Location: `backend/app/agents/research_agent.py`
- Multi-paper analysis
- Gap identification
- Citation network analysis
- Summary generation

2. Code Review Agent

- Location: `backend/app/agents/code_review.py`
- Security vulnerability detection
- Performance optimization suggestions
- Code style enforcement
- Documentation generation

🛠️ Technology Stack

Frontend

- Next.js 14 (App Router)
- Tailwind CSS + ShadCN UI
- Zustand (State Management)
- React Query + Axios
- Monaco Editor
- Recharts + D3.js
- Ethers.js + Wagmi (Web3)

Backend

- FastAPI (Python 3.11+)
- PostgreSQL + SQLAlchemy
- Redis (Cache & Queue)
- Celery (Task Queue)
- FAISS (Vector Store)
- Docker (Sandboxing)
- Pytest (Testing)

AI/ML

- Groq API (Cloud LLM)
- Phi-2 Quantized (Local)
- Sentence Transformers
- LangChain (Agents)
- FAISS (Vector Search)
- Hugging Face Models

Blockchain

- Solidity (Smart Contracts)
- Hardhat (Development)
- Ethers.js (Integration)
- IPFS (Storage)
- Polygon Mumbai (Testnet)
- Chai + Waffle (Testing)

Infrastructure

- Docker + Docker Compose
- GitHub Actions (CI/CD)
- Sentry (Error Tracking)
- Prometheus (Monitoring)
- Winston (Logging)
- Nginx (Proxy)

Deployment

- Vercel (Frontend)
- Railway/Render (Backend)
- Supabase (Database)
- Supabase Storage (Files)
- Cloudflare (CDN)
- Let's Encrypt (SSL)

🚀 Development Phases

Phase 1: Local Development (Weeks 1-8)

Goals:

- Set up development environment
- Implement core features offline
- Integrate local Phi-2 model
- Build basic UI components
- Create Docker sandbox
- Develop authentication system
- Implement file storage
- Build vector store with FAISS

Deliverables:

- Working local development environment
- Basic chat interface with local LLM
- Document upload and processing
- Code execution sandbox
- User authentication flow

Phase 2: Cloud Integration (Weeks 9-12)

Goals:

- Integrate Groq API
- Set up Supabase backend
- Deploy to Vercel (frontend)
- Deploy to Railway (backend)
- Configure CI/CD pipelines
- Implement monitoring
- Add payment integration
- Launch beta version

Deliverables:

- Deployed frontend on Vercel
- Deployed backend on Railway
- Working payment system
- CI/CD pipeline
- Beta version for testing

Phase 3: Advanced Features (Weeks 13-16)

Goals:

- Deploy smart contracts
- Integrate Web3 features
- Add collaboration tools
- Implement advanced agents
- Optimize performance
- Security audit
- Load testing
- Launch production version

Deliverables:

- Blockchain marketplace integration
- Advanced AI agents
- Performance optimizations
- Security audit report
- Production-ready application

🔐 Security Considerations

1. API Security

- **JWT Authentication:** Secure token-based authentication
- **Rate Limiting:** Tier-based request limits
- **API Key Management:** Secure storage and rotation
- **Request Validation:** Input sanitization and validation
- **CORS Configuration:** Strict origin policies

2. Data Security

- **Encryption:** End-to-end encryption for sensitive data
- **Data Anonymization:** PII removal and masking
- **GDPR Compliance:** Data retention and deletion policies
- **Regular Backups:** Automated backup strategies
- **Access Control:** Role-based permissions

3. Code Execution Security

- **Container Isolation:** Separate Docker containers per execution
- **Resource Limits:** CPU, memory, and time constraints
- **Network Isolation:** No external network access
- **Output Sanitization:** XSS and injection prevention
- **File System Restrictions:** Read-only file systems

4. Blockchain Security

- **Multi-sig Wallets:** Multiple signature requirements
- **Audit Trails:** Immutable transaction logs
- **Gas Optimization:** Efficient contract design
- **Upgrade Patterns:** Safe contract upgrades
- **Private Key Management:** Secure key storage

📊 Scalability Architecture

1. Horizontal Scaling

- **Stateless Services:** All backend services designed to be stateless
- **Load Balancing:** Round-robin and health-check based routing
- **Database Replicas:** Read replicas for query distribution
- **CDN Integration:** Static asset distribution via Cloudflare
- **Auto-scaling:** Container orchestration with scaling policies

2. Caching Strategy

- **Redis Cache:** Session data and frequent queries
- **API Response Cache:** LLM response caching
- **Vector Cache:** Pre-computed embeddings
- **Frontend Cache:** Service worker and browser caching
- **Database Query Cache:** Query result caching

3. Queue Management

- **Celery Workers:** Distributed task processing
- **Priority Queues:** Tier-based task prioritization
- **Dead Letter Queues:** Failed task handling
- **Retry Mechanisms:** Exponential backoff strategies
- **Task Monitoring:** Real-time queue metrics

4. Performance Optimization

- **Database Indexing:** Optimized query performance
- **Lazy Loading:** On-demand resource loading
- **Connection Pooling:** Efficient database connections
- **Batch Processing:** Grouped operations for efficiency
- **Code Splitting:** Reduced initial bundle size

🤝 Collaboration Guidelines

1. Git Workflow

- **Branch Strategy:** Feature branches from develop
- **Naming Convention:** feature/[ticket-id]-description
- **PR Requirements:** Code review by 2 team members
- **Commit Standards:** Conventional commits format
- **Version Tags:** Semantic versioning (v1.0.0)

2. Code Standards

- **TypeScript:** Strict mode enabled
- **Python:** PEP 8 compliance
- **Linting:** ESLint + Prettier for JS/TS
- **Testing:** Minimum 80% coverage
- **Documentation:** JSDoc/Python docstrings

3. Documentation

- **API Docs:** OpenAPI/Swagger specification
- **Component Library:** Storybook for UI components
- **Architecture Decisions:** ADR format
- **Onboarding Guide:** Step-by-step setup
- **Wiki:** Technical and business documentation

4. Team Structure

- **Frontend Team:** UI/UX implementation
- **Backend Team:** API and services development
- **AI/ML Team:** Model integration and optimization
- **DevOps Team:** Infrastructure and deployment
- **Blockchain Team:** Smart contract development
- **QA Team:** Testing and quality assurance

5. Communication

- **Daily Standups:** 15-minute sync meetings
- **Sprint Planning:** 2-week sprint cycles
- **Code Reviews:** Within 24 hours
- **Documentation:** Updated with each PR
- **Slack Channels:** Team-specific communication

🔧 Development Scripts

Setup Scripts

```
# Install all dependencies
./scripts/setup/install-deps.sh

# Setup local LLM (Phi-2)
./scripts/setup/setup-local-llm.sh

# Initialize database with migrations
./scripts/setup/init-db.sh

# Setup development environment
./scripts/setup/dev-env.sh
```

Development Scripts

```
# Start full development stack
./scripts/dev/start-dev.sh

# Run frontend only
./scripts/dev/start-frontend.sh

# Run backend only
./scripts/dev/start-backend.sh

# Seed test data
./scripts/dev/seed-data.sh

# Generate TypeScript types from backend
./scripts/dev/generate-types.sh

# Run tests
./scripts/dev/run-tests.sh
```

Deployment Scripts

```
# Deploy frontend to Vercel
./scripts/deploy/deploy-frontend.sh

# Deploy backend to Railway
./scripts/deploy/deploy-backend.sh

# Deploy smart contracts
./scripts/deploy/deploy-contracts.sh

# Update environment variables
./scripts/deploy/update-env.sh

# Run production build
./scripts/deploy/build-prod.sh
```

💰 Cost Optimization Strategy

Free Tier Utilization

Service	Provider	Free Tier Limit	Monthly Cost
Frontend Hosting	Vercel	Unlimited	$0
Backend Hosting	Railway	$5 credit	$0-5
Database	Supabase	500MB	$0
File Storage	Supabase	1GB	$0
Authentication	Firebase	10K MAU	$0
LLM API	Groq	Free access	$0
CI/CD	GitHub Actions	2000 min/mo	$0
Total Initial Cost			**$0-5**

Scaling Cost Projections

- **0-1000 users:** $0-5/month (free tiers)
- **1000-5000 users:** $50-100/month
- **5000-10000 users:** $200-500/month
- **10000+ users:** Custom enterprise pricing

🎯 Key Success Metrics

Technical Metrics

- **Response Time:** < 200ms API response
- **LLM Latency:** < 2s for AI responses
- **Uptime:** 99.9% availability
- **Error Rate:** < 0.1% failed requests
- **Code Coverage:** > 80% test coverage

Business Metrics

- **User Acquisition:** 1000 users in first month
- **Conversion Rate:** 5% free to paid
- **Churn Rate:** < 10% monthly
- **MRR Growth:** 20% month-over-month
- **User Satisfaction:** > 4.5/5 rating

Feature Adoption

- **Chat Usage:** 80% daily active users
- **Code Assistant:** 60% weekly usage
- **Document Q&A:** 40% monthly usage
- **Notebook:** 30% weekly usage
- **Blockchain Features:** 10% adoption

📌 Conclusion

Engunity AI represents a comprehensive, scalable, and innovative approach to combining AI capabilities with blockchain security. The modular architecture ensures:

- **Flexibility:** Easy addition of new features and modules
- **Scalability:** Horizontal scaling capabilities built-in
- **Cost Efficiency:** Leveraging free tiers for initial deployment
- **Security:** Multiple layers of security including blockchain
- **Collaboration:** Clear structure for team development

Next Steps

1. Set up development environment following the setup scripts
2. Begin Phase 1 implementation with core features
3. Establish CI/CD pipelines early in development
4. Create initial UI mockups and user flows
5. Set up project management tools and communication channels

Contact & Resources

- **Documentation:** docs.engunity.ai
- **API Reference:** api.engunity.ai/docs
- **GitHub:** github.com/engunity-ai
- **Support:** support@engunity.ai

Ready to Build the Future of AI-Powered Research?

This architecture provides the foundation for a revolutionary platform that combines the power of AI with the security of blockchain technology.

Download Complete PDF